GDI4-Blatt Thorsten Wink Teil Berechenbarkeit

Es gibt verschiedene Berechenbarkeitsmodelle. Nach der Churchschen These sind die folgenden Modelle äquivalent zum intuitiven Berechenbarkeitsbegriff.

primitiv-rekursiv = LOOP => WHILE = GOTO = TM = RAM = µ-rekursiv

Turingmaschine

T = (A, S, an, •) (Bandalphabet, Zustandsmenge, Anfangszustand, Programm)

Das Programm schreibt man in folgender Art: (s_0, 0) : (s_1, (0, S)), wobei die TM in diesem Beispiel in Zustand s_0 beim Lesen einer 0 in den Zustand s_1 wechselt, eine 0 druckt und den Kopf nicht bewegt.
Eine Mehrband-TM ist immer auch mit einer Einband-TM realisierbar. Dazu teilt man das Band in mehrere Spuren auf und markiert auf jeder Spur, wo sich der entsprechende Kopf befinden würde.

Primitive Rekursion

Die Menge der primitiv rekursiven Funktionen Prim ist folgendermaßen definiert:
Alle folgenden Ausgangsfunktionen sind in Prim:
- die Nachfolgerfunktion
- die Projektion
- die konstante Funktion

Aus primitiv rekursiven Funktionen können neue Funktionen gebildet werden, die ebenfalls in Prim sind durch:
- Substitution
- Primitive Rekursion
- beschränkte Summation
- beschränkte Produktbildung
- Definition durch Fallunterscheidung
- Iteration
- beschränkter • -Operator $(m^b f)(y, x_1,\dots,x_r) = \begin{cases} Min\{z \le y \mid f(z, x_1,\dots,x_r) = 0\}, & falls\ dieses\ Minimum\ existiert \\ y+1, & sonst \end{cases}$

Einige Hilfsfunktionen, um zu zeigen, dass eine Funktion prim.rekursiv ist:
|bin(x)| = floor($\log_2$(x)+1) (Länge einer Binärzahl)
i-te Binärziffer von x: (x div 2^i) mod 2
add, modifizierte Subtraktion, div, mul, mod, signum, Fallunterscheidung, sqrt, log, Fakultät, Potenz

µ-Rekursion

Die Menge µ-Rek der µ-rekursiven Funktionen ist die kleinste Menge von (eventuell partiellen) Funktionen, die die Ausgangsfunktionen
- Nachfolgerfunktion
- Projektion
- konstante Funktion

enthält und abgeschlossen ist unter
- Substitution
- primitiver Rekursion
- Anwendung des µ-Operators

Ackermannfunktion:
1. $A(0, y) = y+1$
2. $A(x+1, 0) = A(x, 1)$
3. $A(x+1, y+1) = A(x, A(x+1, y))$

Satz von Rice

Sei U∈ P_2 akzeptable Aufzählung von P_1 und M ⊄ P_1 nichttrivial. Dann ist Prog(M) = {i | U_i∈ M} nicht entscheidbar.
Beweisweg: Schaue ob Menge eine Indexmenge ist: Eine Funktion finden die drin ist, und eine die nicht drin ist. => nichttrivial => nicht entscheidbar. Beide f müssen primitiv Rekursiv sein.
Die Funktion darf nicht von i abhängig sein.
Beispiel: M = {i ∈ N | Def(U_i ⊆(1,2,3,..10) ist Indexmenge, da z.B. die Identität (x = x)∈ M aber (x=0 wenn x = 0, undefiniert. sonst) ist nicht in M à nach dem Satz von Rice ist M nicht entscheidbar.

Diagonalisierung

Man sucht eine Funktion, die alle Funktionen der Menge abbildet. Dann konstruiert man aus den Funktionen eine neue Funktion, meist unter Verwendung der Diagonalelemente und zeigt, dass diese nicht in der Aufzählung enthalten ist à Widerspruch, also ist die Menge nicht entscheidbar. Das Verfahren verwendet man, wenn die gegebene Menge keine Indexmenge ist.
1. Annahme M sei entscheidbar, dann ex. die charakteristische Funktion xM (und sie ist berechenbar).
2. bilde eine Fkt. f so, dass f und U_i sich in der Diagonalen U_i(i) unterscheiden, dieses f ist nach 1 PREK.
3. es gibt ein j ∈ N mit f = U_j nach Annahme
4. Uj(j) = fj(j) nach 3 und != Uj(j) à Widerspruch, Annahme falsch: M nicht entscheidbar

Beispiel: M = {i ∈ N | Ui(i) Primzahl}
1. Annahme: M entscheidbar, damit existiert xM = {1, x ∈ M; 0 sonst}
2. f(i) := {4, falls xM(i) = 1 (Primzahl); 2 sonst}
3. Es gilt also (Vi∈ N) f(i) != U_i(i).
4. Aber f∈ P_1 nach Annahme, daher (ex. j∈ N) f = U_j
5. (4) = Uj(j) = f(j) != (3) = Uj(j), Widerspruch, daher M nicht entscheidbar.

Zurückführen auf ein bekanntes Problem

Man kann versuchen, eine Entscheidbarkeitsfrage M als einen speziellen Fall eines schon bekannten Problems anzusehen. Wenn man dann weis, dass diese Funktion nicht entscheidbar ist, ist es M erst recht nicht.
Bekannte nicht entscheidbare Probleme: Halteproblem, Wortproblem bei Typ0-Sprachen, Postsches Korrepondenzproblem.

Entscheidbarkeit

	Wortproblem	Leerheitsproblem	Aquivalenzproblem	Schnittproblem
Typ 3	ja	ja	ja	ja
det. kf	ja	ja	ja	nein
Typ 2	ja	ja	nein	nein
Typ 1	ja	nein	nein	nein
Typ 0	nein	nein	nein	nein